AF IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

IN RE APPLICATION OF:

RALPH E. SIPPLE ET AL.

EXAMINER: CLIFFORD H. KNOLL

ART UNIT: 2112

APPLICATION NO.: 09/925,592

FILED: 8/8/2001

TITLE: COMMUNAL LOCK PROCESSING SYSTEM FOR MULTIPROCESSOR COMPUTER SYSTEM

TRANSMITTAL OF MAILING BY EXPRESS MAIL OF APPELLANT'S BRIEF TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Commissioner for Patents
Mail Stop Appeal Brief - Patents
P. O. Box 1450
Alexandria, VA 22313-1450

I hereby certify that this Appellant's Brief To The Board of the Patent Appeals and Interferences in the above-identified application, along with any paper referred to as being attached or enclosed, is being deposited with United States Postal Service with sufficient postage as Express Mail on December 29, 2004 Express Mail Label No. EU647806351US.

The Commissioner for Patents is hereby authorized to charge payment of the required processing fee in the amount of $500.00 to Deposit Account No. **19-3790** as set forth in 37 C.F.R. 1.17(c). A duplicate of this sheet is attached.

CHERYL TARTLER
(Print Name)

Cheryl Tartler
(Signature)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	Confirmation No.: **5422**
Sipple, Ralph E., et al.	Docket No. **RA-5416**
Serial No.: **09/925,592**	Group Art Unit: **2112**
Filing Date: **08/09/2001**	Examiner: **Knoll, Clifford H.**

For: **COMMUNAL LOCK PROCESSING SYSTEM FOR MULTIPROCESSOR COMPUTER SYSTEM**

Mail Stop Appeal-Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPEAL BRIEF PURSUANT TO 37 C.F.R. § 41.37

This brief is being filed in support of Appellant's appeal from the rejections of claims 1-4, 8-16, and 18-20 dated 06/09/2004. A Notice of Appeal was filed on 10/06/2004.

The Commissioner for Patents is hereby authorized to charge payment of the required processing fee in the amount of **$500.00** to Deposit Account No. 19-3790. A duplicate copy of this sheet is enclosed.

01/03/2005 CNGUYEN 00000004 193790 09925592
01 FC:1402 500.00 DA

REAL PARTY IN INTEREST

UNISYS CORPORATION is the real party in interest in the present application. The inventors in the present application assigned their interest to UNISYS CORPORATION on August 8, 2001, and this Assignment was recorded in the USPTO on 08/09/2001 at Reel/Frame 012079/0836.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1-4 and 8-14 stand rejected under section 102(b) over a U.S. Patent reference, Varti, Patent no. 5,678,026, hereinafter (Varti). Claims 15, 16, and 18-20 stand rejected under section 102(e) over a U.S. Patent reference Arimilli, Patent no. 6,625,701. Claims 5-7 and 17 are objected to as depending on rejected base claims but found to contain allowable subject matter and as such are not at issue in this Appeal.

STATUS OF AMENDMENTS

A listing of claims is provided in the APPENDIX below. Claims that have never been amended (namely, claims 2-7, 11, and 16-19) are marked as "original." Claims that have been modified by entered amendments (namely, claims 1, 8-10, 12-15 and 20) are marked as "As Amended." No claims were amended after final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER WITH REFERENCES TO THE SPECIFICATION FOR CLAIM ELEMENTS

Claims 1 and 15 are reproduced below in their currently amended form for ease of reference. The main term on which the appeal turns (Communal Software Locks or CSWLs) is highlighted in both claims. All the other claims (except the claims with allowed subject matter) stand or fall on the discussion of this term as applied through these two independent claims. Therefore the other dependent claims at issue are not reproduced nor discussed as such discussion would be merely superfluity.

Independent Claim 1

1. A circuit apparatus associated with a mid-level cache for handling **side-door communal software lock (CSWL)** inquiries in a multiple instruction-processor computer system said computer system having other mid-level caches with similar circuit apparatae associated with said other mid-level caches, said circuit apparatus comprising:

a. inquiry generator for generating said CSWL inquiries, said CSWL inquiries being signals containing either CSWL requests or status reports regarding locked status of a CSWL, and for providing such CSWL requests and status reports to a side door of one of said other mid-level caches,

b. receiving circuit for receiving said CSWL requests and status reports from said similar circuit apparatae associated with said other mid-level caches,

c. Interpreter for reading the signals in a received CSWL inquiry to determine if it relates to a CSWL mapped to said associated mid-level cache and

what each particular lock request function requires for said received CSWL inquiry,

d. CSWL cache memory within said associated mid-level cache for storing CSWLs to which said associated mid-level cache is mapped, and means for determining if a CSWL which is subject to said received CSWL inquiry is present within said CSWL cache memory,

e. comparator circuit for determining a current value of a requested CSWL within said mid-level cache's memory,

f. CSWL inquiry processor for processing said received CSWL inquiry and for generating a response to said received CSWL inquiry,

g. A circuit for responding to a requesting local processor with a status received from a status report.

Claim 1, meaning and support.

These claims (Claim 1 and its dependents) cover the side door communal software lock. This software lock has two limitations: that it is a side door lock and that it is communal. A side door lock is one using a side door, and as the application states, side doors are used for communal locks. (Abstract: "*A side door communications pathway is set up to handle the communal locks separately from the ordinary data transfer pathways through with ordinary software locks get handled.*") Thus, claims 1-15 are all limited to communal software locks using side doors. Communal Software Locks are defined in detail with reference to the specification following the rest of this description of how the elements of these claims are defined.

CSWL inquiries, sub paragraph a's first limitation, are calls for Communal SoftWare Locks. (Background, lines 5-9, page 2 "calls" for locks can be bottlenecked, lines 14, 15, page) The last full paragraph on page 21 and the one bridging pages 21 and 22 identify that a CSWL request can be generated by

something called a Lock Request Generator or by other processors. It is addressing a component, the SLC controller in the preferred embodiment and says, " *It handles communal locks based on either the presence of a flag in the instruction from its processor or because the lock message came from a side door.* ". This subparagraph "a" also provides for handling of "status reports." The first full paragraph on page 22 through to and including the first full paragraph on page 23 describe this status report function and its functioning in context of the preferred embodiment. This same section describes the preferred embodiment for the sub paragraph "b" receiving components.

The interpreter of sub paragraph c is supported by the preferred embodiment item 1400 of Fig. 13, and accompanying description in the paragraph bridging pages 21 and 22. The other functional limitations of sub paragraph c are supported also in this paragraph.

Communal Software Lock memory areas can be set out separately as in Figs. 5 or 8A or together with the rest of the SLC memory area as in Fig. 8B. The memory array is seen in Fig. 9 (COM-LOCK MEM ARRAY). A discussion of this sub paragraph "d" limitation for the preferred embodiment is set out starting on the paragraph bridging pages 19 and 20, continuing through the paragraph bridging page 21. Note also this sentence from lines 21 and 22, page 22: "*Note that in this Fig. 13, the communal lock cache and mapped directory are within the controller, unlike in Fig. 8A. They can be designed to be in either location.*"

Sub paragraph "e" is supported in discussion of the preferred embodiment identifying a compare circuit CMR in lines 18-20 of page 22.

Sub paragraph "f" is seen in the preferred embodiment as item 1410 of Fig. 13.

Sub paragraph "g" is seen in the preferred embodiment as the controller, best described in the preferred embodiment description by the paragraph at lines 15-29 on page 21. The status that is received from the status report sent to this

controller needs gets to the processor requesting the status through the controller. See paragraph bridging pages 23 and 24 and the first full paragraph on page 24.

Claim 15 is the only other independent claim which defines Communal Software Locks. It is reproduced below.

Independent Claim 15

15. A method for handling communal software locks (CSWLs) among a set of controller circuits situated in an associated set of mid-level caches in a multiprocessor computer system wherein each controller circuit is associated to a one of said set of mid-level caches, said method, comprising:

A) receiving a request for a software lock by a one of said set of controller circuits in a receiving one of said set of controller circuits (a receiving controller) from a requester,

B) interpreting said software lock request and if said interpreting yields a determination that said software lock request relates to a CSWL, then:

C) determining if said requested CSWL is mapped to said receiving controller and if mapped to said receiving controller:

1. searching said associated mid level cache for presence of said CSWL in said associated mid level cache,

2. if said requested CSWL is in a storage circuit in said associated mid level cache either:

a. setting said requested CSWL and returning an ownership indicia to said requester, or

b. if said requested CSWL is owned by another, returning a status to said requester,

3. if said requested CSWL is not in a storage circuit within said associated mid level cache:

a. forming a data request for the requested CSWL and sending said data request to said multiprocessor computer system to retrieve the requested CSWL from a current owner

b. receiving said requested CSWL from said multiprocessor computer system and processing said request in accord with sub-step 2, and

D) if said software lock request does not relate to a CSWL, passing said software lock request as ordinary data within said computer system.

Claim 15, meaning and support.

The preamble sets the claim in a multiprocessor system wherein each of the mid level caches has its own controller. The intermediate level or mid level cache (SLC) is defined at for example, page 27, line19, 20. The other figures that mention cache controllers are mentioned in reference to Fig. 13 at page 21, lines 15-17. Fig. 4 shows the connections between each SLC and each other via a regular pathway and a bold lined radial R. Each SLC has a single processor IP0-IP31 associated with it via a first level cache. (A paragraph bridging pages 10 and 11 describe this). Figures 10A-D illustrate the preferred embodiment operations. A brief summary is in the paragraph bridging pages 22 and 23. The rest of the claims elements are generally defined in the rest of page 23 until line 21 of page 24.

Communal Software Locks, defined.

As the recitation of the claims elements clearly indicates, communal software locks are something that exist in the system or method claimed, along

with non-communal or ordinary software locks which are not claimed. Accordingly some weight should be given to the clear delineation the specification makes between the two types of locks. Communal Software Locks are defined most clearly on page 7:

> Communal locks are handled specially and are the subject of this patent. There are very few communal locks but they constitute a very large percentage of the lock requirements for the partition or the system, and therefore deserve the special treatment given here, since by handling them separately and specially, overall partition or system throughput is enhanced.
>
> Communal locks are determined by the operating system. Schedulers and dispatchers that will be called by every software process needing resources of the computer system, shared as a whole, will typically be mapped as communal locks. In accord with our preferred embodiments, Communal locks do not move from SLC to SLC. Every SLC knows which SLCs own which communal locks because each SLC knows the mapping mentioned above. In the preferred embodiment, each SLC has a separate area for the mapping of communal locks to SLCs. Each SLC has separate areas for the directory of communal locks it owns and for the values of the locks themselves. (These last two areas are similar to the directory and cache the SLC has for data). The "Communal" lock flag will direct the hardware to use the mapped caches when a process calls for a communal lock. Most data and locks are not communal and use the existing caching mechanisms; however, as eluded to above, the communal locks represent a disproportionately high percentage of the lock conflicts encountered in actual operation.
>
> Additionally, a non-standard locking to send-the-function-to-the-data method instead of the normally used send-the-data-to-the-function method of organizing processing power in a multiprocessor system can be

employed, preferably just for handling communal lock requests. In such a system, a lock command is sent from the processor to the cache along with the necessary arguments instead of reading the data from memory into the processor, doing the test and conditionally writing the updated information back to cache lock value. This has the effect of reducing the hardware utilization of the memory busses because the system does not have to send the data to the processor to do a lock, rather the cache is asked to attempt the lock and report whether the attempt was successful.

Communal Software Locks are distinguished from ordinary software locks in the paragraph bridging pages 6 and 7:

> In the preferred embodiments there are two kinds of locks: communal and non-communal. Non-communal locks are handled as ordinary data: to update the lock value, the SLC *<intermediate level cache>* must have ownership of the cache line containing the lock. Communal locks are handled specially and are the subject of this patent. There are very few communal locks but they constitute a very large percentage of the lock requirements for the partition or the system, and therefore deserve the special treatment given here, since by handling them separately and specially, overall partition or system throughput is enhanced.

Since there is nothing in the references that meets this definition for communal software locks, the Patent Office has attempted to shoe-horn the Appellant's claims into a pseudo-common sense definition of software locks that are somehow "communal" as a common, non-technical dictionary might define "communal." This is accomplished by applying hindsight with this incorrect definition in mind to the use of this term "communal" in conjunction with other

defining parts of the term "communal software locks" (also abbreviated as "CSWL") in the claims.

Accordingly we reproduce a common definition of "communal" here for ease of reference, since the rejections do not do so, although the Interview Summary document does have a statement of definition. The Interview Summary document defines "communal" as "characterized by collective ownership and use of property."

From Merriam Webster Online we have this set of definitions for this single word:

Main Entry: **com·mu·nal**
Pronunciation: k&-'myü-n$^{&}$l, 'käm-y&-n$^{&}$l
Function: *adjective*
Etymology: French, from Late Latin *communalis,* from Latin *communis*
1 : of or relating to one or more communes
2 : of or relating to a community
3 a : characterized by collective ownership and use of property **b :** participated in, shared, or used in common by members of a group or community
4 : of, relating to, or based on racial or cultural groups

It is important to note that this definition *does not* refer to software or locks. Neither the examiner nor the applicant appear to have found "communal" relating to software or locks as a term in the known literature prior to this patent application. Thus, there is no accepted meaning in the art of the term Communal Software Locks prior to the definition created in the specification of the application currently under review.

Having identified where in the application the issues for appeal can be found that are present in the claims, the Appellant proceeds to the discussion of the issues at hand.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-4 and 8-14 stand rejected under section 102(b) over a U.S. Patent reference, Varti, Patent no. 5,678,026, hereinafter (Varti). Claims 15, 16, and 18-20 stand rejected under section 102(e) over a U.S. Patent reference Arimilli, Patent no. 6,625,701. Claims 5-7 and 17 are objected to as depending on rejected base claims but found to contain allowable subject matter and as such are not at issue in this Appeal.

ARGUMENT

Issue 1: ***Can an inventor can define its own terms, and use common terms mixed with technical terms to define a claim limitation consistent with the way those terms are used in the specification?***

It is an accepted rubric of patent practice that, "The specification acts as a dictionary when it expressly defines terms used in the claims or when it defines terms by implication." *Guttmann, Inc. v. KopyKake, Inc.* 65 USPQ 2d 1302, 1307 (Fed. Cir. 2002), citing *Vitronics Corp. v. Conceptronic, Inc.* 30 USPQ 2d 1573, 1977 (Fed. Cir. 1996).

In the application, the inventor has defined the term Communal Software Locks to have a specific meaning as described above. While "Communal" may be a term that alone has a common definition, the inventors have combined it

with a technical set of terms, Software Locks, to give the three word term a unique meaning that does not appear to be found in the art.

Further, "[a] definition of a claim term in the specification will prevail over a term's ordinary meaning if the patentee has acted as his own lexicographer and clearly sets forth a different definition." *3M Innovative Properties Company and Minnesota Mining and Manufacturing Company v. Avery Dennison Corporation*, No. 03-1203 (Fed. Cir. Dec. 2, 2003). See also Bell Atlantic Network Services Inc. v. Covad Communications Group Ind,, 262 F.3d 1258, 59 USPQ2d 1865(Fed Cir. 2001) which held that the consistent use of a term in a specification to mean one thing can overcome ordinary meanings of a term.

Finally, see *Irdeto Access Inc., f/k/a TV/Com International Inc., v. Echostar Satellite Corp. n/k/a Echostar Satellite LLC*, Fed Cir, No. 04-1154, 9/14/04 for the proposition that a broad, general meaning of a term that otherwise has no accepted meaning in the art is trumped by a narrower definition in the patent specification.

In the Appellant's application, the term Communal Software Locks is defined as identified above. They are specific function locks, different from ordinary or non-communal locks. These CSWLs handle high volume, often repeated functions (such as schedulers and dispatchers) and their number and which such functions they operate to facilitate may be determined by the operating system. Each one of these CSWLs have a fixed location known by all the intermediate level cache controllers as mapped to a specific intermediate level cache in a computer system with a plurality of such intermediate level caches. This use of the term is consistent throughout the specification and the claims. While there may be questions about whether the mapping is fixed as

partitional boundaries change, (such as during reconfiguration of the computer system or its partitions, or during a reboot) there is no question but that during the life of a partition having multiple intermediate level caches that this mapping does not change. Accordingly, this definition set out clearly by the specification and used consistently throughout should prevail over one cobbled together using a non-technical definition of one of the elements (communal) of the phrase (Communal Software Lock).

Issue 2: ***If a rejection is founded upon a finding that a claim element is in the cited art only by virtue of a mistaken definition of the claim term for that element, does the rejection satisfy the requirements for a valid rejection?***

In not one of the "discoveries" of CSWLs in the art cited by the examiner is there any single mention in those citations that identify that the indicated software lock of the reference is different from ordinary software locks.

In not one of the findings of CSWLs in the art cited by the examiner is there any single mention in those citations that the CSWL is known by other intermediate level caches to be mapped to a specific one of a plurality of particular caches, whether it be the one considering the CSWL request or some other intermediate level cache. The examiner does however point to Vartti, saying....

"In fact, Vartti discloses this in the form of "lock unit 22" seen in Figure 4. "each of the Lock Registers [of Lock Unit 22] holds an address that is locked by the respective requester or for which the requester is waiting for a lock, and information for granting a lock to another requester waiting for a lock on the specified address (col. 10, lines 52-56). This is interpreted to read on "reading the signals in a received CSWL inquiry to determine if it relates to a CSWL

mapped to said associated mid-level cache,; namely, the lock registers are determinative of the association as recited." *(Examiner's Response to Arguments, Advisory Action, bottom of second paragraph.)*

Tit is respectfully submitted that this argument by the Examiner is merely confusing. It equates a mapped area for CSWLs to an area within a lock unit that is not mapped to any particular cache. Instead the mapped area in the reference happens to reside *for the moment* in a particular lock unit associated with a particular cache. Thus, the examiner's statement has the argument nearly exactly backwards.

The Vartti reference teaches that one can have two Storage Controllers, each of them having two separate priority circuits, allowing lock requests to be given high priority while minimizing the impact on general read and write requests. Thus one of the priority circuits controls access to main memory (called Shared Memory 10 in Vartti). As Vartti teaches: "Because of the replicated implementation of the Lock Registers (*which are in the two priority circuits*), a lock request always results in a request to the respective Remote Out Priority circuit 44 or 46, whereas general cache requests do not always result...(*in such requests to 44 or 46*). *(Paragraph from line 36 to line 55 of col. 7, Vartti)* Therefore, in Vartti, ALL LOCK REQUESTS have access to main memory, unlike the requests for CSWLs which only access intermediate level cache memory areas in the invention. This means that Vartti treats all lock requests the same ab initio, thus they cannot be Communal Software Locks as defined by the specification of the Appellant.

In Vartti, there is no mapping of any particular kind of lock to any particular cache. In column 7 lines 10-19, Vartti indicates how locks are granted in parallel, but where they are located is dependent on the requestor (i.e., the instruction processor) and the physical tie of the requestor to a given storage controller. This is the reverse of the Appellant's invention where the mapping of the CSWLs is

dependent on a fixation or assignment to a cache during the life of a partition or system. Here is the language from the Appellant's specification that demonstrates support for this statement:

"..., the system initialization function (the same system initialization function that determines which processors and memory ranges are available to this particular "partition" of the system (one or more partitions may be supported) defines address bits associated with SLC ownership for mapping the address of a communal lock to the SLC *(intermediate level cache)* that owns it, and this mapping exists throughout the life of the set-up. A mapping can be had less preferred ways, such as with a dedicated memory area or other hardware or software, but for all the preferred embodiments, this mapping must be available to all processors (caches) such that each SLC knows where a given communal lock cache line should reside." *(Lines 6-13, page 6, specification)*

Thus in Claim 1 where the Appellant's claim limitations to CSWL's that are enabled by inquiries to determine if the CSWL is mapped to an associated cache is something that by the nature of CSWLs is unavailable in Vartti. Here is that limitation from claim 1:

> c. Interpreter for reading the signals in a received CSWL inquiry to determine if it relates to a CSWL mapped to said associated mid-level cache and what each particular lock request function requires for said received CSWL inquiry,
>
> Also, the mapping itself is claimed:
>
> d. CSWL cache memory within said associated mid-level cache for storing CSWLs to which said associated mid-level cache is mapped, and means for determining if a CSWL which is subject to said received CSWL inquiry is present within said CSWL cache memory,

Thus, Vartti has nothing that corresponds to these limitations because it does not teach anything about Communal Software Locks.

Instead the "mapping" that the Examiner interprets into the wording in Vartti is a transitory presence of a lock in a local lock unit within a storage controller, there because the requestor asked for it and the requestor is associated with that storage controller. And, the storage controller of Vartti is holding a non-communal lock, because it is not known by other storage controllers to be located in this one, instead it is known by this one to be here because this one's requestors asked for it.

But the Examiner has gone further still. On page three of the Advisory Action he states, "That a software lock is communal does not necessarily require them to be not movable, as Vartti clearly demonstrates." Thus it is glaringly clear that the Examiner has relied on his improperly presumptive definition of Communal Software Locks to prove the proposition he makes in the first place. If Vartti had communal locks they would not be as defined in the Appellant's specification because they would be moveable. Thus the Examiner admits that Vartti's locks are different from the locks of the Appellant's invention, without recognizing his admission.

Before leaving this point, the Appellant draws the attention of the Board to the limitations of claim 15 part C in which the operations of the controller for an intermediate level cache are detailed. Here the Appellant's claims limit the function to one in which CSWLs ownership by a cache is known, and to how requests to that owning cache are handled. The Examiner suggests that Arimilli returns ownership in the same way, *(Final Rejection, page 4, penultimate paragraph)* but Arimilli is talking in the cited section about ownership by a processor or requestor. Clearly, again, the Examiner's definition of the Communal Software Lock is used to confuse what is taught by a reference with what is claimed by the Appellant.

The Final rejection also states that "... the cited passage, similarly interpreted to Vartti, is deemed anticipatory of this feature, namely ownership at each second level cache is determined." *(Page 3 of Advisory Action).*

The Appellant perhaps cannot better respond than by indicating that no Communal Software Locks with mapping to specific caches for the life of a system is found in Arimilli, and that this statement by the Examiner clearly shows that the same set of incorrect assumptions exists in citing Arimilli as it demonstrably did in the citation of Vartti against the Appellant's claims.

Issue 3: ***Must the cited art meet all of the limitations of the claims to have a valid rejection under section 102?***

The requirements for a prima facie case of anticipation under section 102 are very clear. Each and every element of the claim must be in the reference, either explicitly or inherently. Anticipation under 35 U.S.C. § 102(e) requires that "each and every element as set forth in the **[**5]** claim is found, either expressly or inherently described, in a single prior art reference."" Verdegaal Bros., Inc. v. Union Oil Co., 814 F.2d 628, 631, 2 U.S.P.Q.2D (BNA) 1051, 1053 (Fed. Cir. 1987).

As has been demonstrated in the earlier portions of this brief, Appellant's claim limitations to Communal Software Locks are not found either expressly or by inherency in the cited references. As such, references to mapping communal software locks to an intermediate cache as defined in support of the definition of Communal Software Locks is also not found in either of the cited references. Thus the cited references do not show communal software locks, nor their

mappings to particular ones of intermediate level caches and are missing essential elements of the claims. Therefore the Patent Office has not made out a prima facie case for a proper rejection.

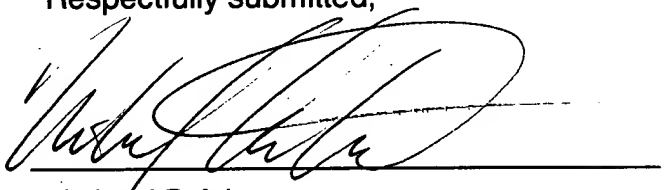
Accordingly, all of the claims should be allowable over the art of record.

CONCLUSION AND PRAYER FOR RELIEF

Neither the Vartti nor the Arimilli have any demonstrable teaching of Communal Software Locks, a key feature of all of the claims at issue herein. The rejection of the claims by assuming that the meaning of the term "communal" was without any limiting value in the claims and could be assumed to be something other than what was carefully defined in the specification was clear error.

Accordingly, Applicant requests that the Board find claims 1-4 and 8-16 and 18-20 allowable, reversing the Examiner's rejection of those claims based on art that does not show communal software locks nor the supporting limitations for them.

Respectfully submitted,



Date: December 27, 2004

Michael B.Atlass
Registration No. 30,606

Unisys Corporation
Unisys Way
M/S E8-114
Blue Bell, PA 19424
Telephone: (215) 986-4111
Facsimile: (215) 986-3090

APPENDIX

Claim 1:

1. (As Amended) A circuit apparatus associated with a mid-level cache for handling side-door communal software lock (CSWL) inquiries in a multiple instruction-processor computer system said computer system having other mid-level caches with similar circuit apparatae associated with said other mid-level caches, said circuit apparatus comprising:

a. inquiry generator for generating said CSWL inquiries, said CSWL inquiries being signals containing either CSWL requests or status reports regarding locked status of a CSWL, and for providing such CSWL requests and status reports to a side door of one of said other mid-level caches,

b. receiving circuit for receiving said CSWL requests and status reports from said similar circuit apparatae associated with said other mid-level caches,

c. Interpreter for reading the signals in a received CSWL inquiry to determine if it relates to a CSWL mapped to said associated mid-level cache and what each particular lock request function requires for said received CSWL inquiry,

d. CSWL cache memory within said associated mid-level cache for storing CSWLs to which said associated mid-level cache is mapped, and means for determining if a CSWL which is subject to said received CSWL inquiry is present within said CSWL cache memory,

e. comparator circuit for determining a current value of a requested CSWL within said mid-level cache's memory,

h. CSWL inquiry processor for processing said received CSWL inquiry and for generating a response to said received CSWL inquiry,

i. A circuit for responding to a requesting local processor with a status received from a status report.

Claim 2:

2. (Original) The circuit apparatus of claim 1 wherein said CSWL inquiry processor is said means for determining of part "d".

Claim 3:

3. (Original) The circuit apparatus of claim 1 wherein said CSWL inquiry processor includes said comparator circuit.

Claim 4:

4. (Original) The circuit apparatus of claim 1 further comprising a CSWL map directory for said associated mid-level cache containing addresses for each CSWL to which said associated mid-level cache is mapped and wherein said CSWL inquiry processor further comprises a circuit for determining whether a CSWL inquiry is mapped to said associated mid-level cache.

Claim 5:

5. (Original) The circuit apparatus of claim 4 further comprising a lock request generator for generating an inter-mid-level cache lock requests to send to other, non-associated mid-level caches if said CSWL inquiry processor determines a CSWL inquiry is not mapped to said associated mid-level cache.

Claim 6:

6. (Original) The circuit apparatus of claim 4 further comprising a lock request generator for generating an inter-mid-level cache lock requests to send a CSWL lock function to the CSWL data determined to be mapped to another, non-associated mid-level cache if said CSWL inquiry processor determines a CSWL inquiry is not mapped to said associated mid-level cache but is mapped to said non-associated mid-level cache.

Claim 7:

7. (Original) The circuit apparatus of claim 5 wherein said lock request generator for generating lock requests determines from information in said CSWL map directory which other, non-associated mid-level cache to which to direct said inter-mid-level cache lock request.

Claim 8:

8. (As Amended) The circuit apparatus of claim 1 further comprising a status stripper circuit for fashioning a signal from a status field in a CSWL after processing by said CSWL inquiry processor to supply information needed to provide a reply to said CSWL inquiry.

Claim 9:

9. (As Amended) The circuit apparatus of claim 1 further comprising a side-door circuit for receiving said CSWL inquiries and status reports from other, non-associated mid-level caches.

Claim 10:

10. (As Amended) The circuit apparatus of claim 8 wherein CSWL inquiries from processor units associated with said associated mid-level cache are received by said circuit apparatus through an internal data channel.

Claim 11:

11. (Original) The circuit apparatus of claim 1 wherein access to said lock cache is given a lower priority than access to a data cache in said associated mid-level cache.

Claim 12:

12. (As Amended) A computer system having a set of mid-level caches wherein said mid-level caches are connected through a side door in each of said mid-level caches to side doors in mid-level caches of other ones of said set of mid-level caches, and wherein a radial communications pathway joins all such side doors to enable side door communication from ones of said set of said mid-level caches to other ones of said set of mid-level caches.

Claim 13:

13. (As Amended) A computer system as set forth in claim 12 wherein the radial is a bus and the connection to each mid-level cache from said radial is said side door of each of said mid-level caches and wherein said side door is programmed to respond only to mapped CSWLs appearing on the bus.

Claim 14:

14. (As Amended) A computer system as set forth in claim 12 wherein the radial is a crossbar and the connections are configured by mapping of said CSWLs such that a given CSWL will map to a unique mid-level cache.

Claim 15:

15. (As Amended) A method for handling communal software locks (CSWLs) among a set of controller circuits situated in an associated set of mid-level caches in a multiprocessor computer system wherein each controller circuit is associated to a one of said set of mid-level caches, said method, comprising:

E) receiving a request for a software lock by a one of said set of controller circuits in a receiving one of said set of controller circuits (a receiving controller) from a requester,

F) interpreting said software lock request and if said interpreting yields a determination that said software lock request relates to a CSWL, then:

G) determining if said requested CSWL is mapped to said receiving controller and if mapped to said receiving controller:

1. searching said associated mid level cache for presence of said CSWL in said associated mid level cache,
2. if said requested CSWL is in a storage circuit in said associated mid level cache either:
   a. setting said requested CSWL and returning an ownership indicia to said requester, or
   b. if said requested CSWL is owned by another, returning a status to said requester,
3. if said requested CSWL is not in a storage circuit within said associated mid level cache:

a. forming a data request for the requested CSWL and sending said data request to said multiprocessor computer system to retrieve the requested CSWL from a current owner

b. receiving said requested CSWL from said multiprocessor computer system and processing said request in accord with sub-step 2, and

H) if said software lock request does not relate to a CSWL, passing said software lock request as ordinary data within said computer system.

Claim 16:

16. (Original) The method of claim 15 wherein step C 2) is performed by said receiving controller in said associated cache and wherein said requested CSWL is retained by said associated mid level cache.

Claim 17:

17. (Original) The method of claim 15 wherein said interpreting of step B comprises; recognizing whether said request has been received into said associated mid level cache through a side door, and if received through said side door then determining that said request is a CSWL request.

Claim 18:

18. (Original) The method of claim 16 wherein said interpreting of step B comprises:

setting a flag indicator by a processor which uses said associated mid level cache to indicate that a software lock request is a CSWL request and recognizing said flag indicator by said receiving controller.

Claim 19:

19. (Original) The method of claim 15 further comprising prioritizing step C wherein step C will be performed by said set of mid level caches only after other mid level cache functions.

Claim 20:

20. (As Amended) The method of claim 15 further comprising prioritizing step C to be performed at a lower priority than at least one other functions of said mid level cache, on an interleaved basis wherein said at least one other function includes but is not limited to transferring data.

Appendix of Cited Art.

Arimili, US Patent No. 6,625,701

Vartti, US Patent No. 5,678,026